The reconstruction of digital holograms on a computational grid

J.J. Nebrensky* and P.R. Hobson
School of Engineering and Design, Brunel University
Uxbridge, Middlesex UB8 3PH, England

ABSTRACT

Digital holography is greatly extending the range of holography's applications and moving it from the lab into the field: a single CCD or other solid-state sensor can capture any number of holograms while numerical reconstruction within a computer eliminates the need for chemical development and readily allows further processing and visualisation of the holographic image. The steady increase in sensor pixel count leads to the possibilities of larger sample volumes, while smaller-area pixels enable the practical use of digital off-axis holography. However this increase in pixel count also drives a corresponding expansion of the computational effort needed to numerically reconstruct such holograms to an extent where the reconstruction process for a single depth slice takes significantly longer than the capture process for each single hologram. Grid computing - a recent innovation in large-scale distributed processing - provides a convenient means of harnessing significant computing resources in an ad-hoc fashion that might match the field deployment of a holographic instrument. We describe here the reconstruction of digital holograms on a trans-national computational Grid with over 10 000 nodes available at over 100 sites. A simplistic scheme of deployment was found to provide no computational advantage over a single powerful workstation. Based on these experiences we suggest an improved strategy for workflow and job execution for the replay of digital holograms on a Grid.

Keywords: holography, digital holography, numerical reconstruction, Grid computing, data extraction

1. INTRODUCTION

Digital holography has the potential to greatly extend holography's applications and move it from the lab into the field: a single CCD or other solid-state sensor can capture any number of holograms while numerical reconstruction within a computer eliminates the need for chemical development and readily allows further processing and visualisation of the holographic image. The steady increase in sensor pixel count and resolution leads to the possibilities of larger sample volumes and of higher spatial resolution sampling, enabling the practical use of digital off-axis holography. However this increase in pixel count also drives a corresponding expansion of the computational effort needed to numerically reconstruct such holograms to an extent where the reconstruction process for a single depth slice takes significantly longer than the capture process for each single hologram. Grid computing - a recent innovation in large-scale distributed processing - provides a convenient means of harnessing significant computing resources in an ad-hoc fashion that might match the field deployment of a holographic instrument.

In this paper we consider the computational needs of digital holography and discuss the deployment of numerical reconstruction software over an existing Grid testbed.

2. DIGITAL HOLOGRAPHY AND GRID COMPUTING

2.1. Digital Holography

Digital holography (strictly, the digital **recording** of holograms) involves replacing the photographic material used in conventional holography with an electronic imaging sensor such as a CCD array. The fringe pattern recorded by the sensor at the moment of exposure is then transferred to a computer and reconstructed numerically (figure 1). The size, shape or the relative position of the dots may then be measured automatically.

* E-mail: henry.nebrensky@physics.org

Holography 2005: Intl. Conf. on Holography, Optical Recording, and Processing of Information,
Yury Denisyuk, Ventseslav Sainov, Elena Stoykova, Eds., Proceedings of SPIE Vol. 6252,
62521I, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.677160

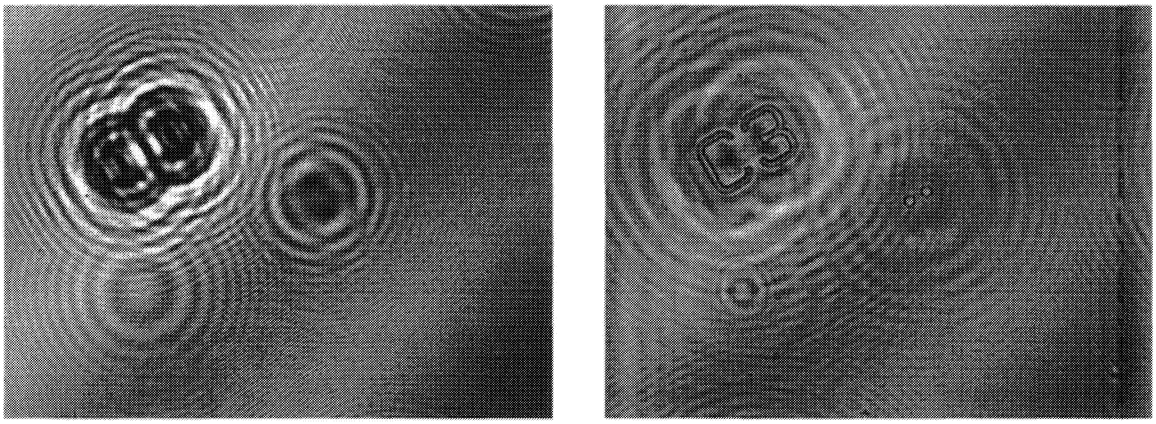

Fig. 1. An in-line hologram of a test target, captured from a CCIR videocamera (*left*) and the in-focus objects (a pair of dots and a label) regenerated within the computer (*right*)(from [1]).

Of course, in useful applications the part of the dots in figure 1 might be played by snowflakes[2], plankton[3] or some other microscopic particles. Unlike a test target, these may be located anywhere through the whole depth of the sample volume. Thus, there is a need to reconstruct a series of slices through the sample volume and then search through them to locate the objects of interest, prior to the individual analysis step. Numerical reconstruction is computationally heavy – multiple FFTs needed for each slice – and intermediate data volumes are huge. The provision of computing power and of data storage are commonly touted as the primary benefits of "Grid computing"; and, since the process of reconstructing any particular slice is independent of all the other slices, hologram replay is an example of an "embarrassingly parallel" application that should be well suited to the Grid.

2.2. Grid Computing

Although there is currently widespread discussion about Grid computing, it is often unclear exactly the term means, especially compared to the established term "distributed computing". By "The Grid" here we mean an infrastructure that allows general purpose computing (rather than being limited to analysing extra-terrestrial emissions or finding prime numbers); supports the notion of a "Virtual Organization" (VO) that controls how its sub-set of the total Grid resources are to be shared among its members; and is geared towards wide-area deployment, with heterogeneous resources spread across the globe. This is similar to the vision originally proposed by Foster and Kesselman[4, 5] and since implemented by them as the Globus Project middleware[6].

A number of Grids and Grid testbeds have been implemented; we refer here to the model followed by the EU DataGrid[7] project and since deployed as the LHC Computing Grid (LCG)[8]. Unfortunately the field is presently encumbered by constantly changing acronyms; hence the terminology here is a personal selection.

The Grid job life-cycle is as follows. At a computer that acts as a User Interface (UI), the user describes the job to be run: the executable, data files to be processed and other requirements (e.g. operating system, replay depth) are specified using JDL (Job Description Language). When the job is submitted, the UI client gathers the JDL and required files into a "sandbox" and passes this to a Resource Broker (RB), which assesses the job requirements and current state of Grid resources, and accordingly identifies the best place to send the job. The Grid has two basic types of resource. A Computing Element (CE) provides the CPUs that do the actual processing – jobs can only run on a CE. A Storage Element (SE) provides mass storage space in which programs and data files can be stored; thus for example rather than repeatedly sending identical copies of an image file in the sandbox to the RB we could instead upload the data once on to a suitable SE, and then refer to this as the data source in the JDL. The RB can then try to choose a suitable CE close (in network terms) to the stored data to run the job. "Best place," in this context, refers not only to physical attributes such as unused disk space left or the correct CPU architecture, but also that the resource supports the user's particular VO. A CE itself consists of a Gatekeeper (GK) that receives the job, and a set of Worker Nodes (WN) that do the actual work – similar to a traditional batch farm. Once the WN has processed the job, the output can be retrieved by the UI and the results are available for use.

We have previously discussed the Grid and how it might be used for the reconstruction of digital holograms[1]. To gauge the utility of the Grid, we have placed a sample hologram on the SE associated with the BITLab facility at Brunel University. HoloPlay was then run on resources around the Grid, with the replayed images being stored on the BITLab SE. The time taken to relative to reconstruct a volume could then be compared to that taken by a desktop PC.

3. EVALUATION OF 'THE GRID'

3.1. HoloPlay submission

We have our own FFT-based reconstruction software "HoloPlay" for single image planes from in-line holograms, which reads all its settings in from a configuration file at runtime. This has been tested with images from an Atmel Camelia camera (8 Mpixel, 12-bit depth) [1]. The same source code compiles and runs both on Visual C++ v. 6 on Windows 2000 and on GCC v. 3.2.2 on Linux (RedHat 7.3). It incorporates the FFTW[9] (v. 3.0.1) and iniParser[10] (v. 2.14) libraries. For this work we compiled without any optimisations.

Our sample holograms are 2300 by 3500 pixels with 16-bit depth. HoloPlay uses PGM format image files, which can be up to 40Mb. Both the hologram and replayed images are thus gzipped before being uploaded to the SE (this then also provides a convenient means of confirming data integrity). To save resources on the RB, the HoloPlay executable itself was also compressed and stored on the SE. The job submitted to the Grid is thus just a shell script wrapper accompanied by a HoloPlay settings file (the JDL for an example HoloPlay submission is given in listing 1). When this script arrives and is run at a WN, it first downloads and unpacks the hologram and the executable, then it runs HoloPlay which reconstructs a slice from the hologram according to the settings file (`HoloPlay.ini`), and finally the script compresses the slice image and uploads it to the SE at Brunel University.

```
[    VirtualOrganisation = "holoplay";
    Executable = "holoplay_wrap.sh";
    Arguments = "00";
    StdOutput = "holoplay.out";
    StdError = "holoplay.err";
    InputSandbox = {
        "holoplay_wrap.sh",
        "HoloPlay.ini"
    };
    OutputSandbox = {
        "holoplay.out",
        "holoplay.err"
    };
    RetryCount = 1;
    requirements = ( other.GlueHostOperatingSystemName=="Redhat"
                            || other.GlueHostOperatingSystemName=="RedHat" );
    JobType = "normal";
    Type = "Job"]
```

Listing 1. An example JDL file for submitting HoloPlay jobs to a Grid.

This process is shown diagrammatically in figure 2: the UI sends the requests for each slice sequentially to the RB, which forwards each job to some CE as it arrives. The jobs then run and upload their slice to the SE independently (asynchronously). The UI must poll the RB for job status and error messages – thus in principle the UI need not be connected to the network while the jobs are running. The standard output and error streams from HoloPlay are captured by the WN and are returned to the UI as files named `holoplay.out` and `holoplay.err` respectively.
The result from the first run of just one slice is shown in figure 3. We believe this to be the first use of a large-scale Grid testbed for the replay of a digital hologram.

To evaluate the utility of the Grid, we have submitted batches of 10, 20 and 40 single slices, and looked at how long it takes between starting the submission and the replayed images arriving at the SE ready for further processing (from their filesystem timestamp). As can be seen in figure 4, there is a delay of more than five minutes before any jobs complete, partly due to the overhead of the Grid infrastructure. Completed images then accumulate at a steady rate until the last few slices remain outstanding. These have suffered various delays in the system and dribble in over a period of hours. The performance of the Grid can be compared with replaying slices sequentially on a single computer. Figure 4 includes the rates of reconstruction on two typical machines. "Reference D" shows the work rate of the computer provided by

BITLab as a WN on its CE (and indeed used by some of the Grid-run jobs): this has a 1500 MHz Pentium 4 CPU and 512 MB of memory, and takes ~340 s to run a single invocation of HoloPlay on our test hologram. It can be seen that the Grid can deliver replayed slices much faster than a single such machine. The second reference machine, "F", is the desktop PC also used as the UI and has an AMD Athlon XP 2800+ Model 10 CPU (actually 2080 MHz) and 2 GB of dual-channel memory. Although this CPU is only ~50% faster than "D", it reconstructs a slice from the test hologram in only 75 s and it can be seen from figure 4 that it can out-pace the Grid single-handed for modest numbers of images, while for a batch of 40 slices it lags behind in the medium term but eventually catches up as the Grid sits waiting for the return of stray and aborted jobs.Although disappointing, the Grid's relatively poor performance is hardly surprising; rather, it confirms that "the Grid" is NOT some magic bullet that satisfies all computing needs, but merely just another tool that must be wielded correctly to gain useful results. Our usage of the Grid in this work has been very unsophisticated; we will next describe some of the shortcomings in the present workflow and suggest some solutions.

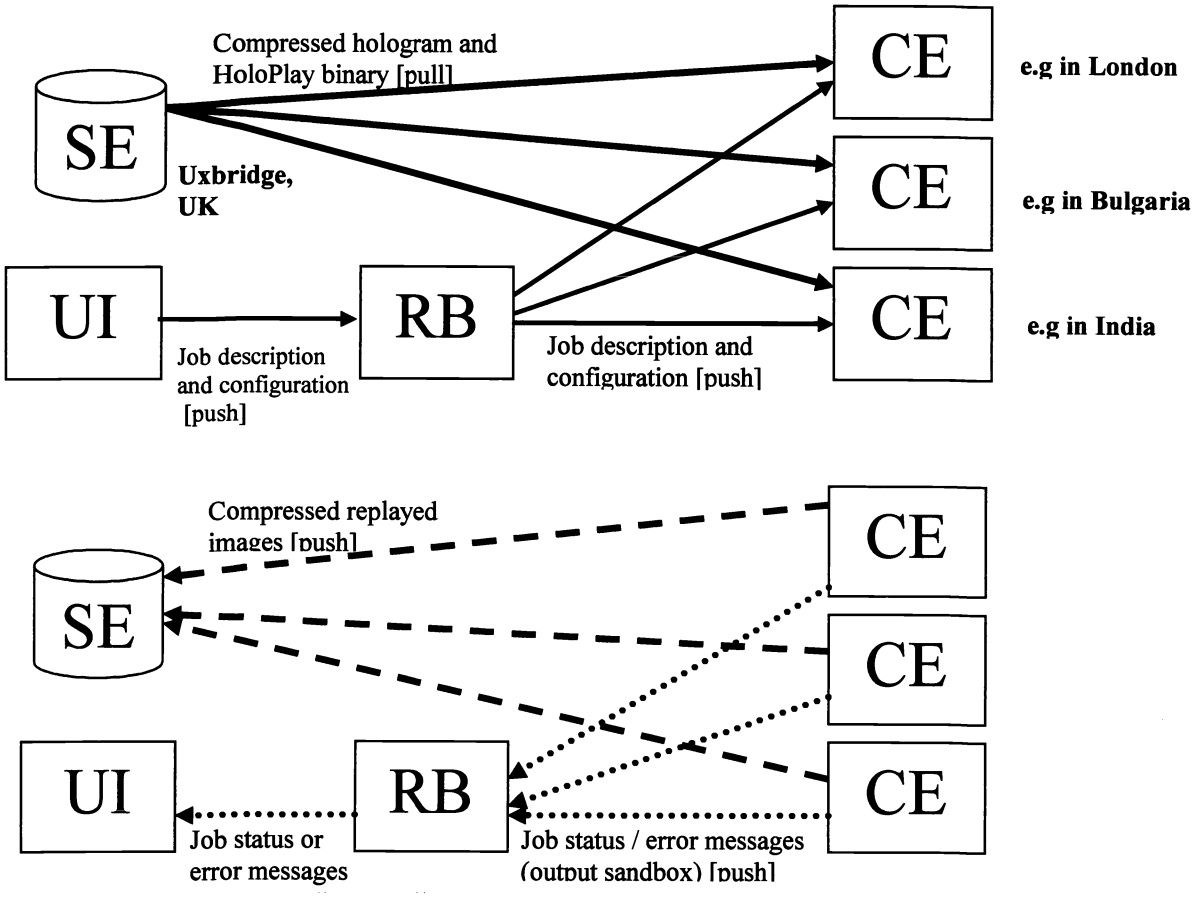

Fig. 2. Reconstruction of hologram sample volume on Grid: job submission (*top*) and completion

One noticeable issue is that one computer "F" carries out the task nearly five times quicker than the other, "D". We believe this is because FFTW – which is very memory-hungry – exhausts D's physical memory and starts to swap. Given that D was actually used by the Grid for running HoloPlay jobs, it is reasonable to suppose that many of the nodes at other sites were also unsuitable for this particular task (indeed, as Red Hat Linux 7.3 is already unsupported, it seems likely that newer, more powerful resources will actually be using some other operating system). Clearly there should be a requirement in the JDL that the WN used for replay must have sufficient physical memory.

Another problem is that for remote jobs the overheads in job submission and data transfer of the hologram take nearly the same time as the task itself. The Grid would be used much more efficiently if each remote job replayed and uploaded several image slices – this would simply need a loop in the shell script wrapper. Network connectivity at remote sites can

also cause problems; e.g. in prior testing it took 20 minutes to transfer a reconstructed 40 MB image back from a resource in India. This is just one possible reason why a small proportion of jobs take significantly longer than average to return their results. In the long term, network connectivity is expected to become a factor that the RB takes into account when deciding where to send jobs. There can also be issues with the data bandwidth to/from disk on the SE – if too many remote jobs simultaneously upload results, the seeking of the hard drive heads between separate files can slow data transfers down to a fraction of the effective rate when the files are transferred sequentially. The compromise between the maximum number of simultaneous connections allowed by an SE and the risk of data transfers stalling unpredictably should be re-examined on any dedicated Grid testbed where the size and frequency of data transfers are well-defined. The final problem is that currently LCG job efficiency is just over 90% – i.e. 10% of jobs fail completely (never return data) because of mis-configured sites or middleware problems. The `RetryCount` key in the JDL used here (listing 1) allows the Grid infrastructure to resubmit failed jobs itself a given number of times, but it can take over 12 hours for stalled jobs to be noticed and sent elsewhere. It must also be pointed out that replayed images are not returned in any particular order: thus just because 30 out of 40 image slices are available on the SE does **not** mean that a contiguous 75% of the sample volume is ready for further analysis… There is thus a need to track the progress of submitted jobs and the location of the completed image files so that missing slices can be filled in quicker. We have previously[1] identified BOSS[11] as a possible solution; the simple nature of the reconstruction task means that parameter-sweep frameworks such as APST[12] also deserve investigation.

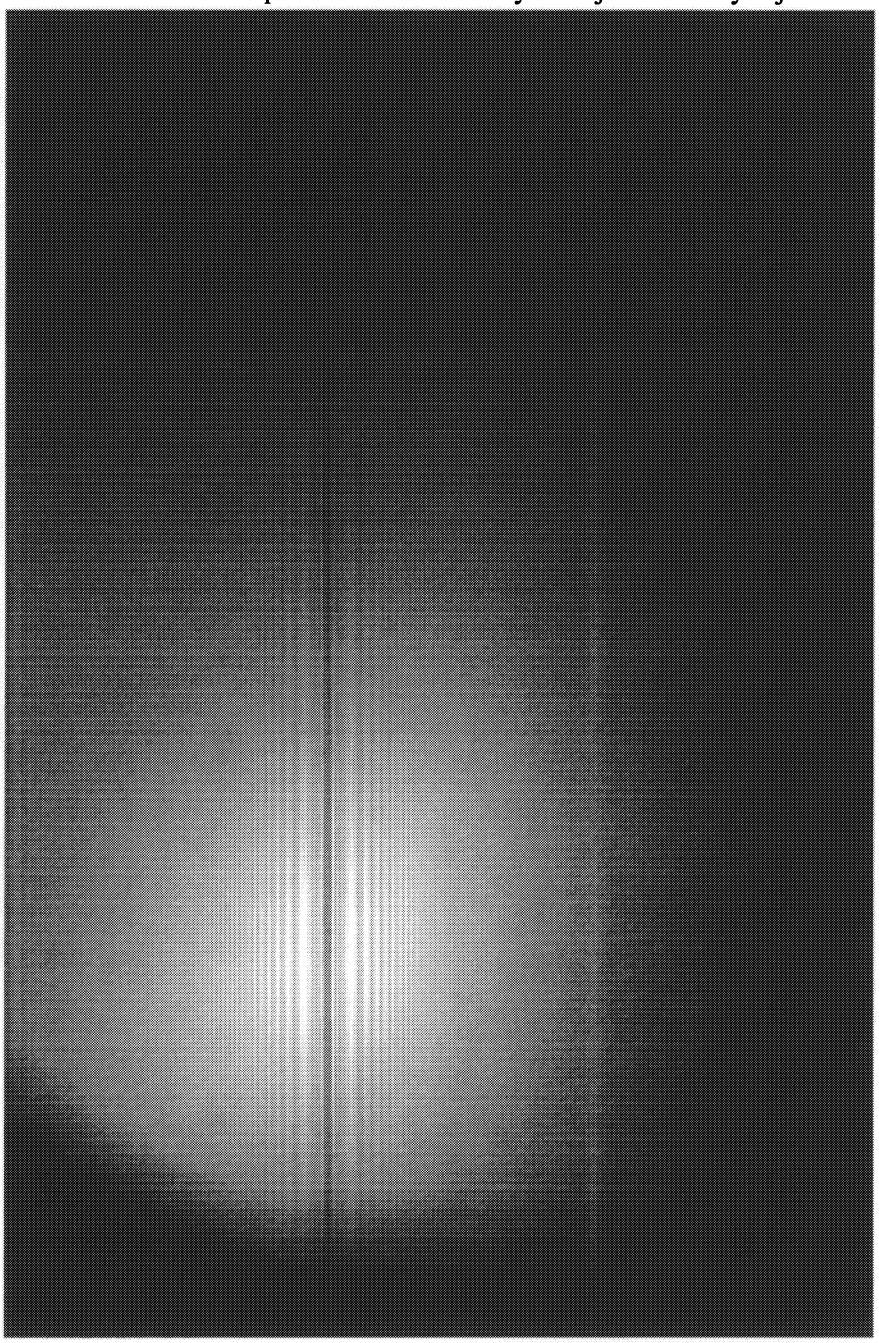

Fig. 3. First known hologram replayed using WAN Grid: an out-of-focus reconstruction of a 50 μm wire.

(*bottom*).

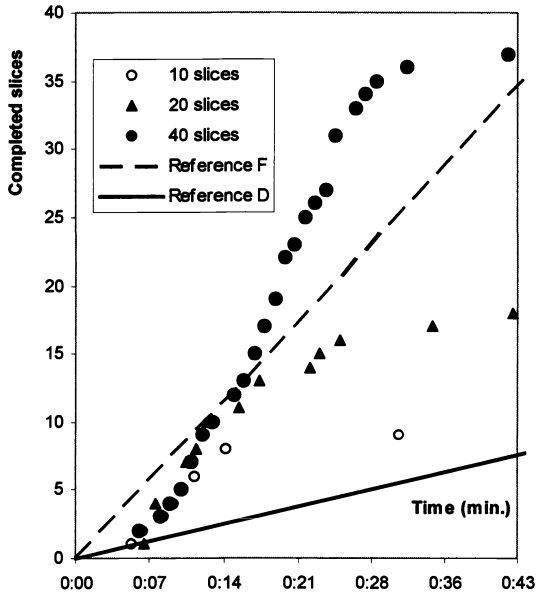

Fig. 4. Cumulative completion rate of HoloPlay jobs on the Grid and on standalone PCs.

CONCLUSIONS

Improvements in sensors and image processing are making possible new applications for digital holography, but demand ever-larger sensors and faster frame-rates, and hence greater computing resources for volume reconstruction. We believe that Grid computing will provide an effective and convenient way to meet the computing needs of novel holographic instruments, but to make effective use of such resources it is still necessary to understand both the needs of the computational jobs themselves and also their interactions with the Grid infrastructure. We plan now to investigate whether the solutions we have suggested above will help the Grid realise its potential benefits for our application.

It should also be kept in mind that Grid computing technology offers holography potentially useful options for secure data transfers and storage in addition to its number-crunching capabilities.

ACKNOWLEDGEMENTS

We would like to thank Marc Fournier-Carrié and Paul Fryer for their work with us in creating the HoloPlay software, and the CMS Collaboration for providing access to LCG resources.

REFERENCES

1. J.J. Nebrensky, P.R. Hobson and P.C. Fryer: "Grid computing for the numerical reconstruction of digital holograms" in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments III*; Ryszard S. Romaniuk, Ed. *Proc. SPIE* vol. **5775** pp. 285-296 (2005)
2. H-J. Vössing, S. Borrmann and R. Jaenicke: "In-Line Holography of Cloud Volumes Applied to the Measurement of Raindrops and Snowflakes" *Atmospheric Research* **49** pp.199-212 (1998)
3. P.R. Hobson and J. Watson: "The Principles and Practice of Holographic Recording of Plankton" *Journal of Optics A: Pure and Applied Optics* **4** pp. S34-S49 (2002)
4. I. Foster and C. Kesselman: "*The Grid: Blueprint for a New Computing Infrastructure*" 2nd revised Ed. Morgan Kaufman (2003)
5. I. Foster, C. Kesselman and S. Tuecke: "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" *International Journal of High Performance Computing Applications* **15** (3) pp. 200-222 (2001) [available online: http://www.globus.org/research/papers/anatomy.pdf]
6. The Globus Alliance. On-line [available http://www.globus.org/]
7. The EU DataGrid. On-line [available http://www.eu-datagrid.org/]
8. The LHC Computing Grid. On-line [available http://lcg.web.cern.ch/LCG/]
9. FFTW. On-line [available http://www.fftw.org/]
10. N. Devillard: "iniParser" (2003) [available on-line]
11. C. Grandi and A. Renzi: "Object Based System for Batch Job Submissions (BOSS)" *CMS Note 2003/005*, (2003); On-line [available http://www.bo.infn.it/cms/computing/BOSS/]
12. H. Casanova and F. Berman: "Parameter Sweeps on the Grid with APST" Chapter 33 in "*Grid Computing - Making the Global Infrastructure a Reality*" eds. F. Berman, A. Hey and G. Fox; Wiley ISBN: 0-470-85319-0 (2003)